Automatic Network Configuration with Dynamic Churn Prediction

Andri Lareida, Thomas Bocek, Maxat Pernebayev, Burkhard Stiller
University of Zurich
Department of Informatics (IFI)
Communication Systems Group (CSG)
Zurich Switzerland
Email: [lareida—bocek—stiller]@ifi.uzh.ch, maxat.pernebayev@gmail.com

***Abstract*—Peer-to-Peer (P2P) systems have been deployed on millions of nodes worldwide in environments that range from static to very dynamic and therefore exhibit different churn levels. Typically, P2P systems introduce redundancy to cope with loss of nodes. In distributed hash tables, redundancy often fixed during development or at initial deployment of the system. This can limit the applicability of the system to stable environments or make them inefficient in such environments. Automatic network configuration can make a system more adaptable to changing environments and reduce manual configuration tasks. Therefore, this paper proposes an automatic replication configuration based on churn prediction that automatically adapts its replication configuration to its environment. The mechanism termed dynamic replication mechanism (dynamic RM) developed and evaluated in this paper is based on exponential moving averages to predict churn that is used itself to determine a replication factor meeting a certain reliability threshold. Simulations with synthetic data and experiments with data from torrent trackers show that the behavior can be predicted accurately in any environment, from low churn rates to diurnal and high churn rates.**

I. Introduction

Distributed systems are important to handle large amounts of data and many users. A distributed system consists out interconnected nodes, contributing its resources to process parallelized tasks. Many different types of distributed system exists, ranging from large-scale Internet systems such as P2P applications like BitTorrent or Bitcoin to local cluster systems such as distributed key-value storage or distributed file systems. A key aspect is the management and configuration of such distributed systems.

A common requirement for all types of distributed systems is fault tolerance, which is achieved by applying redundancy. In case of churn, a process which is described by the arrival and departure of nodes, redundancy of resources can compensate for lost resources. However, the redundancy configuration depends on to deployment environment of such a system. In general, redundancy requirements are fundamentally different when deployed in static environments that have high availability in contrast to user controlled home PCs which are turned off when not in use. While in an local cluster environment (e.g. Apache Cassandra) nodes will rarely leave and, therefore, redundancy can be lower compared to a user-controlled environment (e.g. P2P file sharing), where the churn follows a diurnal pattern and the churn rate is much higher.

Redundancy is often configured by a static replication factor (RF) that defines the number of replicas created. However, churn is not constant [19]. Thus, RF has to be high enough to cope with the worst case. This leads to a waste of resources when the churn rate is lower than the worst case. Therefore, this paper proposes dynamic replication mechanism (dynamic RM) that automatically determines and dynamically adapts RF of a P2P system based on dynamic churn rates. This mechanism predicts the future churn rate based on local churn observations and adapts RF accordingly.

The advantages of a dynamic RM include the reduction of overhead through unnecessary replications and the adaptability of the system. With the ability of adapting to changing conditions a system can increase its range of possible deployments to local and global networks without decreasing efficiency. Furthermore, the input values for the dynamic RM is reliability and observation length (e.g. 99.9%, 60 seconds), which makes it easier to configure the distributed system for a specific quality of service.

The remainder of this paper is structured as follows. Section II gives an overview over the related work and compares it the dynamic RM approach. Section III presents the design and explains the churn prediction and replication approach. While Section IV describes the evaluation of the churn prediction and replication approach, Section V summarizes and concludes this paper.

II. Background and Related Work

This section presents the underlying framework used for the prototype implementation. Furthermore, an overview over existing replication mechanisms is given to show where existing solutions differ from our approach. It is important to keep in mind, that the target environment for the prediction and replication mechanism are both Internet *and* local systems. Thus, churn prediction based on diurnal pattern (such as [18]) is not suitable as this only targets Internet environment.

A. TomP2P

TomP2P is a P2P framework, providing a distributed hash table and a tracker [20]. Neighbor peers are stored in a map and listeners can be added to get notified once a new peer joined, or an existing peer left the network. TomP2P is the underlying framework and the prediction is built as a module

on top of it. The source code of TomP2P with the prediction module is available online[1]. TomP2P implements also a replication mechanism which creates copies of data in order to provide high availability. There are two types of replication mechanisms available: direct and indirect replications. In direct replication, an originator peer is responsible for refreshing replicas periodically. This peer periodically checks if there are enough replicas or not. When the originator peer goes offline, the replication process stops and all replicas disappear when TTL (time to live) expires. In indirect replication, the closest peer to the content will be responsible for periodically checking if enough replicas exist. Therefore, the originator peer can go offline at any time. In case a new peer joins the system and it is the closest peer to content, it will become responsible. Whenever a responsible peer leaves the system, the responsibility will be delegated to the peer which is the next closest. In both replication types, the replication factor is automatically set by the dynamic RM.

B. Replication

One of the inherent characteristics of P2P systems is the presence of churn [19]. Various authors suggest to minimize the churn which can be reached with sophisticated neighbor selection mechanisms as described in [5], [7]. Other authors suggest [11] to place replicas on more reliable nodes that can be found with availability prediction. However, churn remains and it is important to replicate data not to lose it. Selected replication mechanisms are described in this Section.

Within the **Symmetric Replication** scheme, each identifier is associated with a set of f distinct identifiers in the system. As a result, there will be $\frac{N}{f}$ equivalence classes [4]. Nodes who are responsible for items of one identifier of equivalence class store items of other identifiers of that equivalence class. It implies that in order to find an item with identifier i, any identifiers associated with i can be requested. For example, the identifier 0 is associated with identifiers 4, 8 and 12, in identifier space of 16. So any node who is responsible for any of these identifiers should store all of them. As a result, any of those nodes responsible can be asked to find an item with identifier 0.

In **Successor-List Replication** keys are replicated to the k immediate successors [17]. This approach is simple and requires only nodes knowing their k immediate successors, respectively. The RelaxDHT [8] approach can be used in key-based routing DHTs (Distributed Hash Table), such as Pastry or Chord. The condition of replicating a key to the k immediate neighbors is relaxed to replicating a key to k nodes in the responsible nodes' leaf set (neighbor set). Therefore, migration of data can be avoided as long as a sufficient number replicas exist in a leaf set. The replication factor k is still a static constant and cannot be determined automatically.

ID-Replication [17] eliminates the drawbacks of a successor list replication. Instead of assigning an identifier to a node identifiers are assigned to groups and node identifiers are only unique inside a group. Therefore, a group is responsible for a key range and not a single node. All nodes of a group replicate every key in the group's responsibility. If the group size falls below a threshold or above a limit management protocols split or merge groups.

Chordet [12] is a replication mechanism for Chord. Chordet distributes replicas evenly in the logical ring. This is achieved by using ID-generating, which can considerably reduce the lookup failure rate and the lookup path length. A key advantage is that not all nodes have to be employed to benefit from its effects. Even if a node is not aware of a replication, it will benefit from it if the lookup path contains a node which runs Chordet.

Multiple Identity Replication eliminated one drawback of ID-Replication, which is the split and merge operations required with groups. A self-stabilizing algorithm based on multiple identities can be achieved that does not require any cooperation between nodes [15]. When a node joins the network it creates i identities, where $i > 1$. This node can shut down identities based on locally observed activities. If a specific key is requested rarely, the identity close to this key will be shutdown and a new random identity will be created. Thus, more nodes will be around popular content, while less popular content will have a smaller node density. Such an algorithm can run without coordination, however, it will virtually increase the churn rate.

Table I compares these replication mechanisms with respect to the following dimensions: adaptive replication rate adaption, replica placement, and local decision taking.

TABLE I: Overview of related replication mechanisms.

	Adaptive Replication	Replica Placement	Local Decisions
Multiple Identity	×	✓	✓
ID-Replication	×	×	×
Successor-List	×	✓	×
Symmetric	×	×	✓
Dynamic RM	✓	×	✓

Churn prediction is often used to place replicas on nodes that stay longer online. While this replica placement problem and *diurnal* churn prediction based on churn models with user-controlled nodes have been investigated extensively [2], [22], only little work exists in the area of *generic* churn prediction and adaptive replication rate. Although diurnal pattern prediction allows for better churn prediction as reported in ([16], [18], [1]) and works well in global P2P systems, in local cluster systems, such a behavior pattern cannot be exploited.

Furthermore, our approach, dynamic RM, adapts dynamically the replication parameter based on information from local churn observations only. Thus, if a node can make decisions based on local observations it is not dependent on other nodes and does not need any information from other nodes, making the system more robust.

Although mechanisms exist to automatically set parameters based on observed network conditions as shown by [9], where the routing table size is set automatically based on the available bandwidth, or based on diurnal pattern, where user behavior is predicted, none of those replication mechanisms automatically configures the replication factor for generic churn that runs in both Internet *and* local systems.

[1] https://github.com/tomp2p/TomP2P

III. Basics and Design of Dynamic Replication

To dynamically determine the replication factor (RF), churn has to be known or at least estimated. For the estimation, moving averages are used, which will smoothen the data and also show a trend that can be used for short term prediction. Moving averages are used *e.g.* for stock market prediction or forecasting sales [21]. Furthermore, if churn is changing a good prediction of future churn will improve the accuracy of RF. Pure P2P systems do not show an entity having a complete overview of the entire system, such as for [10]. However, any node makes local observations such as neighbors joining or not being reachable anymore. Based on these observations every node can make churn predictions. This prediction will be used to calculate the RF.

This section presents the basics used to predict churn and it explains the design of the algorithm to calculate the RF based on a desired reliability value. Finally, some insights about the implementation are given.

A. Predicting Churn

Every node has to make observations of node arrivals and departures. Here, an observation x is made during a time interval Δt, e.g. 20 minutes. The number of observations used to calculate the prediction m is called Observation Length (OL) and denoted k. OL influences the prediction: the lower the OL the more responsive the prediction is to fluctuations in churn; a higher OL results in a smoother, but less responsive prediction. For the churn prediction, moving averages are used, which are often applied in financial problems, e.g. predicting or outlining stock prices. Different alternatives of moving averages [3] are described in the following.

The **Simple Moving Average** (SMA) is calculated as the regular average except that only the k last observations x of churn at time t are considered instead of all observations made. Therefore, SMA changes with every new observation. The SMA is defined by the formula:

$$sma_t = \frac{x_{t-k+1} + x_{t-k+2} + ... + x_t}{k}$$

The larger the value k the smoother SMA will become; if $k = t$ SMA becomes the Simple Average (SA). The lower k gets the more responsive SMA becomes to changes in these observations. If $t < k$, SMA cannot be calculated, because k observations could not be made yet. However, the SA can be used as a substitute until t reaches at least k.

The **Simple Weighted Moving Average** (SWMA) applies weights on the on the k last churn observations x. SWMA at time t can be defined as follows:

$$swma_t = w_1 x_{t-k} + w_2 x_{t-k+1} + ... + w_k x_{t-1}, \quad \sum_{i=1}^{k} w_i = 1$$

This is only a general description and does not give an answer on how to distribute the weights among the observations.

The **Exponential Moving Average** (EMA) considers recent events as more important than older events [3]. Therefore, it could react on recent changing behaviors while a certain smoothing effect on the churn prediction will be maintained. Another advantage of EMA is that it does not require all observations to be stored, since it can be calculated from the last EMA and the new observation x_t at time t, as shown in the formula:

$$ema_t = ema_{t-1} + \alpha(x_t - ema_{t-1}), \quad t > 1$$

The smoothing factor α can be calculated by the formula:

$$\alpha = \frac{2}{k+1}$$

Since k is the observation length, the higher k gets the less influence an observation gets and the smoother the curve becomes. The closer α approaches 1 the more responsive the result becomes. If $\alpha = 1$ $ema_t = x_t$ and it does not determine an average anymore.

The **Dynamic Exponential Moving Average** (DEMA) uses linear regression analysis to determine the OL $\hat{k}$, which gives the best linear fit, where $3 \leq \hat{k} \leq k$. For each possible value of $\hat{k}$ the regression line with the least squares and the coefficient of determination, denoted R^2 [13], is calculated. The closer R^2 is to 1, the better the regression line fits. The OL which gives the highest value for R^2 is used for $\hat{k}$.

B. Replication Factor

The RF depends on three parameters: the desired reliability r of the system, the number of nodes that will leave the system m, and the number of neighbor nodes n. r expresses the probability that a record is kept safe in a value between 0 and 1. The reliability has to be defined by the P2P application developer. m is the output of the moving average based prediction. n is known by any node participating in a DHT and is used to determine the (local) rate of nodes leaving the system.

Let p be the probability of RF nodes being within the predicted number of departing m nodes, then RF can be defined as the smallest number which fulfills the requirement $1 - p >= r$, where p is the probability of RF nodes being among the m nodes predicted to depart. Therefore, $1 - p$ determines the probability of RF nodes not being among the nodes predicted to depart. Automatic replication is supposed to provide a minimal reliability of r and, therefore, $1 - p$ must be greater or equal than r.

To calculate p, a group of nodes leaving the system is considered. The probability p of node $P1$ being among the nodes leaving the system is $\frac{m}{n}$. Therefore, the probability p of $P1$ (p_1) and $P2$ (p_2) being among the leaving nodes is

$$p = p_1 * p_2 = \frac{m}{n} * \frac{m-1}{n-1}$$

Accordingly, the probability of RF nodes leaving the system can be expressed as:

$$p = p_1 p_2 \dots p_{RF} = \frac{m}{n}\frac{m-1}{n-1}\dots\frac{m-RF+1}{n-RF+1} = \prod_{i=0}^{RF-1}\frac{m-i}{n-i}$$

To find a replication factor that fulfills this condition, RF can be increased while repeatedly calculating the reliability until it is greater or equal to the desired reliability.

C. Implementation

The dynamic replication has two hooks in TomP2P. The first hook is to get notified when a peer joins or leaves the network (PeerMapChangeListener). This information is stored in a circular buffer since the complete history is not required and resources can be spared. The buffer serves as an input for the prediction model. The second hook is the replication executor that is periodically called. The implementation default to call the replication executor is 60 seconds, which allows to react in a reasonable time to changing churn rates and does not induce too much load. That means every 60 seconds the new replication factor RF is calculated to satisfy the reliability r defined by the application developer. Hence, the prediction model needs to predict the churn rate for the next 60 seconds. The details of the implementation can be found on Github [2].

IV. Evaluation

To investigate the performance of dynamic RM, a thorough evaluation has been conducted. First, the churn prediction models from Sec III are investigated by feeding them with real life churn data, calculating their predictions, and comparing them to each other. Second, the most suitable model is tested in an experiment running a DHT and introducing churn based on synthetic and real data while the data loss is measured.

A. Data

The evaluation is based on two data sets defining the churn rate over time. The first is a synthetic data set that can be used for simulating churn in a local system. The second data set is based on data collected from BitTorrent trackers [6]. Since a measurement study [19] found that the overall dynamics of node participation in content-sharing systems is similar to the dynamics in DHTs, it is valid to use the BitTorrent churn data to evaluate dynamic replication in DHTs.

The BitTorrent data [6] was gathered by monitoring four different torrents and nodes connected to their swarms. Trackers were queried every 20 minutes over 7 days. The data collected contains IP addresses and port numbers of nodes, which can be uniquely identified by this information. Therefore, churn rates can be calculated with a resolution of 20 minutes. The four torrents monitored and their initial swarm sizes are shown in Table II.

TABLE II: Key parameters of the BitTorrent data set.

ID	Torrent Description	Initial Swarm Size
T_1	Under the Dome S01E04 480p HDTV	639
T_2	Falling.Skies.S03E07.HDTV	1338
T_3	Defiance S01E11 720p HDTV	1175
T_4	Orange.Is.The.New.Black.S01E10.720p	900

B. Comparison of Prediction Models

The first model for churn prediction evaluated is **SMA**, since it is a very simple approach and it is expected to be the least performing. **EMA** serves as a second model, which is a more complex version of SMA. The last model used is **DEMA** which is the most complex one. The evaluation will show if DEMA is worth the computational overhead.

The ideal value for the Observation Length (OL) needs to be determined, which tells how many observations are considered to predict the churn rate. OL influences the smoothness of the prediction curve. A smaller OL means that the prediction fluctuates more with changes of the churn rate, which can lead to a lot of overhead due to too many changes of the replication factor. A higher OL means that the prediction curve will become smoother and therefore RF might not be adjusted although it would be necessary.

Figure 1 shows real churn observations and predictions made by the DEMA model for OL 10, 30, and 60. The x-axis represents the time in minutes and the y-axis indicates the number of nodes leaving the system. The solid blue line, labeled data set, depicts the actual observation from data set T_1. The dashed lines show the predicted values with different OLs. Only a part of the entire data set is shown, because in this case the difference between the lines becomes better visible. An OL of 10 means that no more than 10 observations are used to calculate the prediction of how many nodes will leave in the next time interval. The Figure shows that a lower OL results in a more responsive curve.

To determine the effects of OL on the prediction, each prediction model is tested with observation lengths of 5, 10, 15, 20, 30, 40, 50 and 60. To identify the model with the most accurate prediction and the optimal observation length, the convergence of the predicted churn value to the ideal prediction is analyzed. The more the prediction converges with the measured values, the better the prediction of the model. The convergence is calculated at every time step t and a point for the best model is recorded. After all ts are calculated the relative score (points an OL scored divided by total points awarded) is used to compare the models. As input for the calculations the measured observations from T_1 is used, which serves 503 data points or ts.

Figure 2 shows the results of this convergence analysis. The x-axis represents the percentage of wins, the y-axis steps through the different OLs, and the z-axis is used for comparing different models. As expected the DEMA clearly dominates for OLs below 50. EMA yields, for the two lowest OLs, better results than SMA. However, at OL 50 and 60 SMA performs best but the difference is marginal and can be attributed to the characteristics of the data set since the results are inconsistent, *e.g.* DEMA at OL of 60 is higher than at Ol 50. In general it can be concluded that lower values for the OL lead to better convergence and DEMA converges most of all.

Since the prediction is the basis for the calculation of RF its accuracy is also investigated. The ideal replication factor RF_i is again calculated from those 503 data points of T_1. The model-specific RF_m is calculated on the basis of the same three models with the same OLs and the RF is calculated and compared to RF_i. A RF is considered accurate if $RF_i <= RF_m <= RF_i + 3$, that means there is a tolerance to have a higher than necessary RF but not a lower one. Because a lower RF cannot guarantee the desired reliability there is no tolerance in that direction. The systems reliability values r of 0.90 0.99 and 0.9999 are evaluated. Investigating values below 0.90 is not relevant, since reliability below 90% can be easily

[2]https://github.com/tomp2p/TomP2P/tree/master/replication

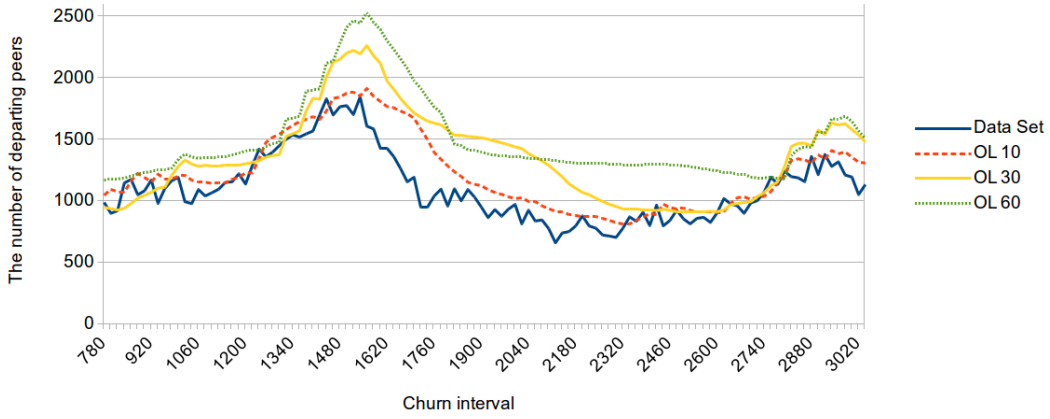


Fig. 1: Effect of the Observation Length (OL) on the prediction of the DEMA model based on data set T_1.

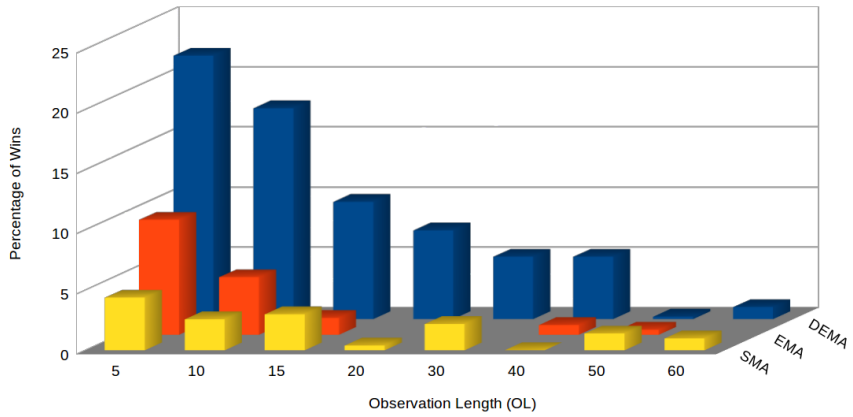


Fig. 2: Convergence of three prediction models

achieved by a small RF, especially in those conditions tested. The other values are chosen to determine the mechanism's behavior when the reliability approaches 1.

Figure 3 shows the results for a reliabilities 0.9 and Figure 4 for 0.9999, which were chosen to show the lower and uper end boundaries. The first difference to the convergence result is that the three models (SMA, EMA, DEMA) are much closer to each other. For EMA and SMA it can be stated that for a high reliability the smaller the OL the more accurate the RF will be. An unexpected observation is seen with DEMA 0.9999 where there appears to be a peak at OL = 10. This behavior can be explained by the increasing sensitivity to small changes with lower OL. Since the time window of events taken into account with OL = 5 is very short, a temporary incline in a generally declining trend can be considered as a trend change and less peer departures will be predicted, what is wrong and leads to a worse performance. With a lower reliability value this effect might not become visible since the threshold for changing the RF is higher. For reliability of 0.9 there is a jump from OL = 5 to OL = 10 for OLs $>$ 10 the improvement is minor. Therefore, the DEMA model with OL = 10 can be considered the generally most accurate model in the conditions tested, which is used in data loss experiments. Further evaluations of the dynamic RM can be found in [14].

C. Data Loss Experiment

To evaluate how much data is lost in a synthetic and the BitTorrent case, an experiment based on the TomP2P DHT is run. For the implementation, the RF was bounded between 2 and 6. The first reason is that the underlying DHT implementation allows only values between 2 and 6 to be configured. The second reason is that a value bellow 2 would result in no replication at all and that temporary bursts in churn changes cannot push the RF beyond reason.

The experiment runs in turns in which a uniformly distributed percentage of nodes is removed and added to the system. This constitutes the worst case in which all nodes leave at the same time during a turn. This is the most critical case which a DHT needs to handle and it allows to speed up the experiment since 20 minutes of churn resolution from the data sets can be simulated in a few seconds without loss of accuracy. Furthermore it allows for checking the number of replicas for each key stored in the DHT every round and thus to discover data losses. 5000 keys are placed into the DHT and the number of participating nodes varies from 639 t 1338

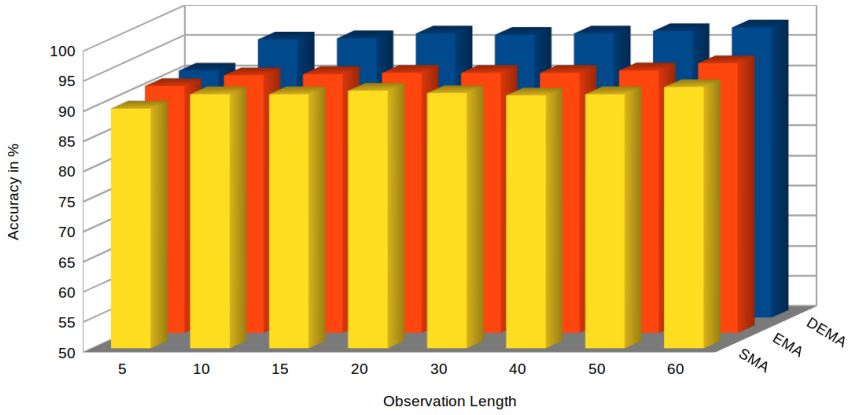


Fig. 3: Replication factor accuracy compared between models and different OLs and for reliabilities 0.9.

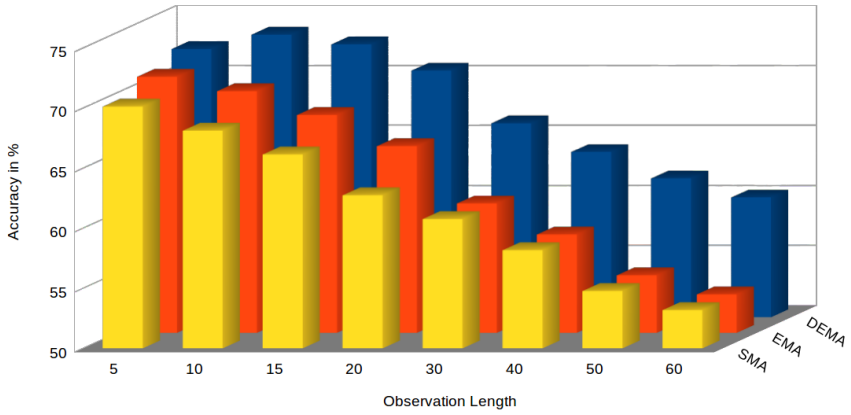


Fig. 4: Replication factor accuracy compared between models and different OLs and for reliabilities 0.9999.

depending on which data set is used T_1 or T_2. The synthetic data sets are labeled as follows:

- *R1* churn fluctuations range of 5% to 10%
- *R2* churn fluctuations range of 5% to 15%
- *R3* churn fluctuations range of 5% to 20%
- *R4* churn fluctuations range of 5% to 30%

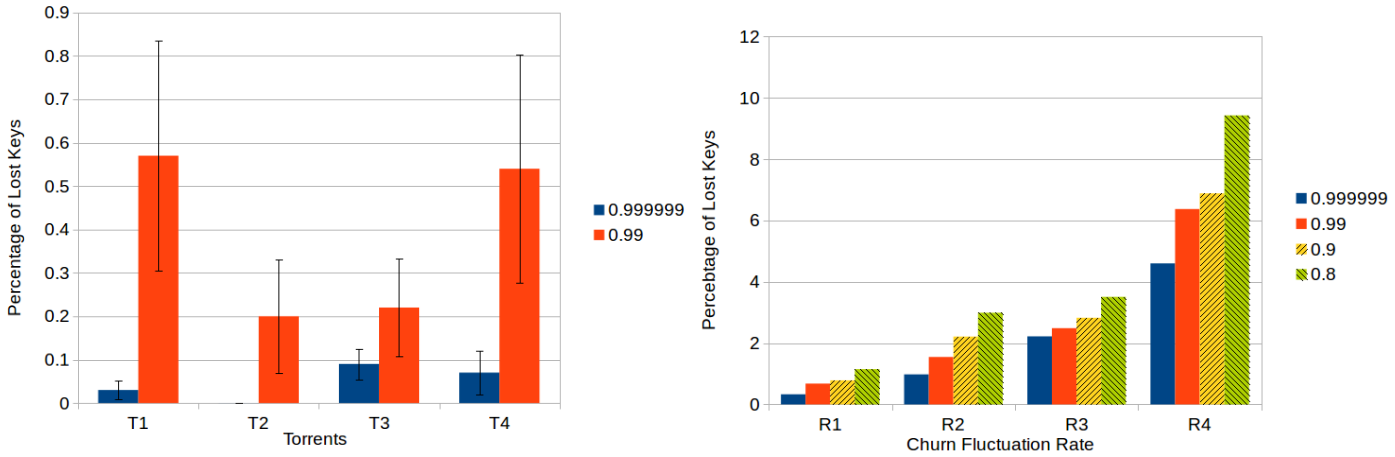
The experiment is conducted on a single machine with two 12 Core AMD Opteron(tm) 6180 SE processors accompanied by 64GB of RAM. Since these experiments ran on a single machine no side effects from the network (i.e. bandwidth limitations of switches) or other experiments are introduced.

Figure 5a shows the percentage of data loss for reliability values of 0.99 and 0.999999 with the error bars showing the standard deviation of three experiment runs to show the variance. The average data loss for all torrents was 0.38% and 0.05% for the reliabilities of 0.99 and 0.999999, respectively. The maximum data loss for the two reliability values was 0.57% and 0.09%, respectively. For 0.99 reliability data loss stays below 1% but there seems to be a lower bound at 0.09%. With increasing reliability data loss is decreased.

Figure 5b shows the percentage of data loss for reliability values of 0.80, 0.90, 0.99 and 0.999999 for the different ranges of churn fluctuation *R1 to R4*. The figure shows that data loss behaves as expected, the more churn the more loss and the higher the reliability value the less loss occurs. However, the data loss is higher than in Figure 5a, which is explained with the worst case settings that were used, where nodes are churning in bursts.

V. Summary, Conclusions, and Future Work

This paper presented the dynamic replication mechanism (dynamic RM) which automatically determines and dynamically adapts the replication factor in a distributed system according to the churn rate. Dynamic RM depends on the prediction of churn which is determined based on locally observed churn rates. Different models with several observation lengths were tested with real life data to get achieve accurate predictions. The dynamic exponential moving average with an observation length of 10 yielded the best churn predictions.



(a) The average data loss based on real life data sets.

(b) The average data loss of synthetic data for one hour.

Fig. 5: Data loss for various reliability values.

The experiment conducted to investigate data loss showed that the system behaves as intended: the replication factor is set dynamically based on the churn rate and the input values: observation length and reliability. If the reliability is increased, the data loss decreases due to a higher dynamically adjusted RF. For both data sets, the same implementation was used and the configuration was done automatically. Thus, dynamic RM can be used in global Internet P2P application as well as local distributed systems without manually configuring the RF.

The evaluation of moving average prediction showed that churn is best predicted by the dynamic moving average model. In a real world Internet scenario, dynamic replication will perform well for reliabilities up to at least 0.9999. It can be concluded that the dynamic replication performed well with the synthetic data set considering that the worst-case scenario was tested.

With the dynamic RM presented here, distributed systems become more flexible and gain adaptability to different environments. The same application can be used in a high availability cluster or in the Internet where availability of peers is much lower without the overhead of using a high redundancy in all cases.

Future work includes further experiments with more nodes and more churn scenarios, comparing dynamic RM to a baseline with perfect churn prediction. Furthermore, availabilities from real data centers will be included in future experiments.

Acknowledgments

This work was supported partially by the SmartenIT and the FLAMINGO projects funded by the EU FP7 Program under Contract No. FP7-2012-ICT-317846 and No. FP7-2012-ICT-318488, respectively.

References

[1] R. Bhagwan, S. Savage, and G. Voelker, "Understanding Availability," in *Peer-to-Peer Systems II*, ser. Lecture Notes in Computer Science, M. Kaashoek and I. Stoica, Eds. Springer Berlin Heidelberg, 2003, vol. 2735, pp. 256–267.

[2] A. Binzenhöfer and K. Leibnitz, "Estimating Churn in Structured P2P Networks," in *20th International Teletraffic Conference on Managing Traffic Performance in Converged Networks*, ser. ITC20'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 630–641. [Online]. Available: http://dl.acm.org/citation.cfm?id=1769187.1769257

[3] R. Brown, *Smoothing, Forecasting and Prediction of Discrete Time Series*, ser. Dover Phoenix editions. Dover Publications, 2004.

[4] A. Ghodsi, L. Alima, and S. Haridi, "Symmetric Replication for Structured Peer-to-Peer Systems," in *Databases, Information Systems, and Peer-to-Peer Computing*, ser. Lecture Notes in Computer Science, G. Moro, S. Bergamaschi, S. Joseph, J.-H. Morin, and A. Ouksel, Eds. Springer Berlin Heidelberg, 2007, vol. 4125, pp. 74–85.

[5] P. B. Godfrey, S. Shenker, and I. Stoica, "Minimizing Churn in Distributed Systems," in *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*. Pisa, Italy: ACM, September 2006, pp. 147–158.

[6] G. M. Jutz, "Scraping bittorrent trackers," Bachelor's Thesis, University of Zurich, 2013.

[7] J. Ledlie, J. Shneidman, M. Amis, and M. Seltzer, "Reliability-and Capacity-based Selection in Distributed Hash Tables," Harvard University Computer Science, Technical Report, 2003.

[8] S. Legtchenko, S. Monnet, P. Sens, and G. Muller, "RelaxDHT: A Churn-resilient Replication Strategy for Peer-to-peer Distributed Hash-tables," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 7, no. 2, pp. 28:1–28:18, July 2012.

[9] J. Li, J. Stribling, R. Morris, and M. F. Kaashoek, "Bandwidth-efficient Management of DHT Routing Tables," in *2nd Conference on Symposium on Networked Systems Design & Implementation (NSDI)*. Berkeley, CA, USA: USENIX Association, July 2005, pp. 99–114.

[10] P. Maymounkov and D. Mazieres, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," in *1st International Workshop on Peer-to-Peer Systems (IPTPS)*, Cambridge, MA, USA, March 2002, pp. 53–65.

[11] J. W. Mickens and B. D. Noble, "Exploiting Availability Prediction in Distributed Systems," in *3rd Conference on Networked Systems Design & Implementation (NSDI)*. San Jose, CA, USA: USENIX Association, May 2006, p. 6.

[12] G. Park, S. Kim, Y. Cho, J. Kook, and J. Hong, "Chordet: An Efficient and Transparent Replication for Improving Availability of Peer-to-peer Networked Systems," in *ACM Symposium on Applied Computing (SAC)*. Sierre, Switzerland: ACM, March 2010, pp. 221–225.

[13] K. Pearson, "On Lines and Planes of Closest Fit to Systems of Points in Space," *Philosophical Magazine*, vol. 2, no. 6, pp. 559–572, 1901.

[14] M. Pernebayev, "Replication, Synchronization, and Automatic Network Configuration in P2P Networks," Master's thesis, University of Zurich, 2013.

[15] A. Rao, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, "Load Balancing in Structured P2P Systems," in *Peer-to-Peer Systems II*, ser. Lecture Notes in Computer Science, M. Kaashoek and I. Stoica, Eds. Springer Berlin Heidelberg, 2003, vol. 2735, pp. 68–79.

[16] K. Rzadca, A. Datta, and S. Buchegger, "Replica Placement in P2P Storage: Complexity and Game Theoretic Analyses," in *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on*, Genoa, Italy, June 2010, pp. 599–609.

[17] T. Shafaat, B. Ahmad, and S. Haridi, "ID-Replication for Structured Peer-to-Peer Systems," in *Euro-Par 2012 Parallel Processing*, ser. Lecture Notes in Computer Science, C. Kaklamanis, T. Papatheodorou, and P. Spirakis, Eds. Springer Berlin Heidelberg, August 2012, vol. 7484, pp. 364–376.

[18] N. Shahriar, M. Sharmin, R. Ahmed, M. Rahman, R. Boutaba, and B. Mathieu, "Diurnal Availability for Peer-to-peer Systems," in *Consumer Communications and Networking Conference (CCNC), 2012 IEEE*, Las Vegas, Nevada, USA, Jan 2012, pp. 619–623.

[19] D. Stutzbach and R. Rejaie, "Understanding Churn in Peer-to-peer Networks," in *6th ACM SIGCOMM Conference on Internet Measurement (IMC)*. Rio de Janeriro, Brazil: ACM, October 2006, pp. 189–202.

[20] The TomP2P Project, "TomP2P, a P2P-based high performance key-value pair storage library," http://tomp2p.net/, last visited: 2014.

[21] P. WINTERS, *Forecasting Sales by Exponentially Weighted Moving Averages*. Carnegie Institute of Tcchnology - Defense Technical Information Center, 1959.

[22] D. Wu, Y. Tian, and K.-W. Ng, "Analytical Study on Improving DHT Lookup Performance under Churn," in *Sixth IEEE International Conference on Peer-to-Peer Computing*, Sept 2006, pp. 249–258.